Recent advances in real geometric reasoning

Davenport, J. and England, M.

Recent Advances in Real Geometric Reasoning

James H. Davenport[1] and Matthew England[2]

[1]Departments of Computer Science & Mathematical Sciences, University of Bath, UK
[2] Department of Computing, Coventry University, UK
J.H.Davenport@bath.ac.uk, Matthew.England@coventry.ac.uk

Abstract. In the 1930s Tarski showed that real quantifier elimination was possible, and in 1975 Collins gave a remotely practicable method, albeit with doubly-exponential complexity, which was later shown to be inherent. We discuss some of the recent major advances in Collins method: such as an alternative approach based on passing via the complexes, and advances which come closer to "solving the question asked" rather than "solving all problems to do with these polynomials".

1 Introduction

Although methods with better asymptotic complexity are known in theory (e.g. [GV88]), the workhorse of implemented algorithms for real geometric reasoning is Cylindrical Algebraic Decomposition. This was introduced in [Col75] to produce a remotely practicable (complexity "merely" doubly exponential in the number of variables) alternative to Tarski's original method from 1930 [Tar51], whose complexity could not be bounded by any tower of exponentials. Tarski in fact set out to solve the *quantifier elimination problem* for real algebraic geometry (Section 4): given $Q_{k+1}x_{k+1}Q_{k+2}x_{k+2}\ldots\Phi(x_1,\ldots,x_n)$, where $Q_i \in \{\forall,\exists\}$ and Φ is a Boolean combination of relations involving polynomials $p_i(x_1,\ldots,x_n)$, find an equivalent $\Psi(x_1,\ldots,x_k)$, where Ψ is a Boolean combination of relations involving polynomials $q_i(x_1,\ldots,x_k)$. In fact, we cannot solve this in the language of algebraic geometry: we need *semi-algebraic* geometry, allowing $>$ as well[1] as $=$. The *necessity* of $>$ follows from the fact of the example $\exists y : x = y^2 \Leftrightarrow (x > 0) \vee (x = 0)$; its *sufficiency* is the point of Tarski's work.

2 Cylindrical Algebraic Decomposition by Projection and Lifting

[Col75] constructs a sampled[2] *Cylindrical Algebraic Decomposition* (CAD) of $\mathbf{R}^n$ which is *sign-invariant* for the p_i, where these words are defined as follows.

[1] Strictly speaking $>$ is sufficient, but implementations always allow $\geq$ and $\neq$. In fact, $\neq$ is intrinsic to the regular chains approach discussed in Section 3.

[2] The "sampled" nature is implicit in [Col75, and its successors], but the authors find it helpful to be explicit about this.

Definition 1 (CAD terminology). *Note that throughout we are ordering our coordinates/variables, so that x_n is the "last coordinate".*

decomposition: *a partition of $\mathbf{R}^n$ into cells $C_{\mathbf{i}}$ indexed by n-tuples of natural numbers (so $\mathbf{R}^n = \bigcup_{\mathbf{i}} C_{\mathbf{i}}$ and $\mathbf{i} \neq \mathbf{j} \Rightarrow C_{\mathbf{i}} \cap C_{\mathbf{j}} = \emptyset$);*

(semi-)**algebraic:** *every $C_{\mathbf{i}}$ is defined by a finite set of equalities and inequalities of polynomials in the x_i, including expressions of the form*

$$\mathrm{RootOf}_2(f_1(x_1, y)) < x_2 < \mathrm{RootOf}_3(f_2(x_1, y)) \tag{1}$$

(where RootOf_2 means "the second real root, counting from $-\infty$");

cylindrical: *for all $k < n$, if π_k is the projection onto the* first *k coordinates, then, for all $\mathbf{i}, \mathbf{j}$, $\pi_k(C_{\mathbf{i}})$ and $\pi_k(C_{\mathbf{j}})$ are either equal or disjoint;*

sampled: *for each cell $C_{\mathbf{i}}$ there is an explicit point $s_{\mathbf{i}} \in C_{\mathbf{i}}$;*

sign-invariant: *for the polynomials in Φ on each cell, every p_i is identically zero, or everywhere positive, or everywhere negative.*

Collins constructed such a decomposition by a process now known (at least by our colleagues) as *CAD by Projection and Lifting* (for more details see [Dav15]). The key property in this approach is the following.

Definition 2. *A polynomial $p(x_1, \ldots, x_m)$ is* delineable[3] *over a region $C \subset \mathbf{R}^{m-1}$ if:*

1. *the portion of the real variety of p that lies in the cylinder $C \times \mathbf{R}$ over C consists of the union of the graphs (called sections) of some $k \geq 0$ continuous functions $\theta_1 < \cdots < \theta_k$ from C to $\mathbf{R}$* **and**;
2. *there exist integers $m_1, \ldots, m_k \geq 1$ such that, for every point $(a_1, \ldots, a_{m-1})$ in C, the multiplicity of the root $\theta_i(a_1, \ldots, a_{m-1})$ of $p(a_1, \ldots, a_{m-1}, x_m)$, considered as a function of x_m alone, is m_i (and in particular is constant).*

A set of polynomials is delineable over C if each is delineable **and** *if the sections are either identical or disjoint. This is actually equivalent to saying that the product is delineable.*

Intuitively, if the $\{p_i\}$ are delineable over C, their graphs neither fold nor cross.

Let $\mathcal{P}_n$ be the set of polynomials in Φ, with coefficients from some effective[4] field $K \subset \mathbf{R}$. Then Collins algorithm proceeds as follows:

[3] There are various, subtly different, definitions in the literature. This one is from [McC99].

[4] The literature often stipulates $\mathbf{Q}$ or the algebraic numbers $\mathbf{A}$. The real requirement is that we can perform all the polynomial algebra we need over K, and that, given expressions $a, b \in K$, we can decide the trichotomy $a < b$ or $a = b$ or $a > b$. Once we start adding transcendental functions to our language, the effectivity of K becomes a major problem, as we run across the usual indecidability results. This is addressed in different ways in [AMW08] and [Vor89,Vor92].

Projection: Given some $\mathcal{P}_k \subset K[x_1, \ldots, x_k]$ construct a set $\mathcal{P}_{k-1} \subset K[x_1, \ldots, x_{k-1}]$ such that, over each cell of a CAD sign-invariant for $\mathcal{P}_{k-1}$, the polynomials of $\mathcal{P}_k$ are delineable. Though the details depend on the algorithm, the key ingredients are leading coefficients (where these vanish some θ_i tends to infinity), discriminants (where these vanish some θ_i ceases to have constant multiplicity) and resultants (where these vanish, the θ_i from different polynomials intersect).

Repeat until we have the set of univariate polynomials $\mathcal{P}_1$.

Base case: Given $\mathcal{P}_1$, isolate the N_1 real roots of these polynomials in $\mathbf{R}^1$, and construct a CAD consisting of the N_1 roots, and the $N_1 + 1$ intervals between them (or to the left/right of them all). The sample points for the 0-dimensional cells are the roots themselves: for the 1-dimensional intervals we choose any convenient point, generally rational and with denominator the smallest power of 2 we can find.

Lifting: Given a CAD D_{k-1} of $\mathbf{R}^{k-1}$, sign-invariant for $\mathcal{P}_{k-1}$, construct a CAD D_k of $\mathbf{R}^k$, sign-invariant for $\mathcal{P}_k$. For each cell $C_{\mathbf{i}}$, this is done by substituting the sample point $s_{\mathbf{i}}$ into $\mathcal{P}_k$, and doing the equivalent of the base case for the resulting univariate system (valid across the whole of $C_{\mathbf{i}}$ if the projection operator provides delineable projection polynomials).

Repeat until we have the CAD D_n of $\mathbf{R}^n$.

If we suppose that $\mathcal{P}_n$ contains m polynomials, of degree (in each variable) bounded by d, and coefficient length bounded by l (coefficients bounded by 2^l), then the time complexity is bounded [Col75, Theorem 16] by

$$O\left(m^{2^{n+6}}(2d)^{2^{2n+8}}l^3\right). \tag{2}$$

This analysis is very sensitive to the details of the sub-algorithms involved, and a more refined analysis of the base case [Dav85] reduces the complexity (though not the actual running time) to

$$O\left(m^{2^{n+6}/4}(2d)^{2^{2n+8}/6}l^3\right).$$

This improvement might seem trivial, but in fact implies taking the fourth root of the m, d part of the complexity.

A less sensitive property (and one that reflects the cost of *using* such a decomposition) is the number of cells: for the Collins method this is bounded, by an analysis similar to [BDE+14], by

$$O\left(m^{2^n}(2d)^{2\cdot 3^n}\right). \tag{3}$$

As is often the case in mathematics, we get more insight if we solve an apparently harder problem. [McC84] did this, demanding that the decompositions D_k, $k < n$ be, not just sign-invariant, but

order-invariant for the polynomials in Φ, i.e. on each cell, every p_i is identically zero, and vanishes to the same order throughout the cell, or everywhere positive, or everywhere negative.

This actually lets his $\mathcal{P}_k$ be much simpler than Collins', with the cost that the lifting procedure might fail if some element p of $\mathcal{P}_k$ *nullifies* (is identically zero) over some cell in D_{k-1}. In this case, McCallum says that $\mathcal{P}_k$ was not *well-oriented*, and has to either:

1. proceed by working around the problem or concluding it not relevant. This is only possible in certain cases (e.g. the cell is dimension 0) [Bro05]. Otherwise;
2. revert to Collins' projection (or a variant due to [Hon90]); or,
3. add the partial derivatives of p to $\mathcal{P}_k$ and resume the projection process from there — an operation that to the best of the authors' knowledge has never been implemented, doubtless because of the complicated backtracking involved, and the fact that, whereas we only *ought* to add this polynomial in the nullifying region, the design of Collins' algorithm and its successors assume a global set of polynomials at each level.

"Randomly", well-orientedness ought to occur with probability 1, but we have a family of "real-world" examples (simplification/branch cuts, see [BBDP07]) where it often fails. The analogy of (3) is given by [McC85, Theorem 6.1.5] as

$$O\left(m^{2^n}(2d)^{n\cdot 2^n}\right), \tag{4}$$

and a recent improved analysis in [BDE+14, (12)] reduces this to

$$O\left(2^{2^{n-1}}m(m+1)^{2^n-2}d^{2^n-1}\right). \tag{5}$$

3 CAD by Regular Chains

This alternative to the traditional computation scheme of projection and lifting was introduced in [CMXY09], then improved in [CM14a]. The method can be described as "going via the complexes", since the authors first construct a cylindrical decomposition of $\mathbf{C}^n$, and then infer a CAD of $\mathbf{R}^n$. They make use of the well developed body of theory around regular systems [Wan00] for the work over the complexes, and the algorithms are all implemented in the `RegularChains` Library[5] for Maple, hence our designation: *CAD by Regular Chains*.

We first need the following analogue of Definition 2 (not precisely analogous, as Definition 2 allows for non-square-free polynomials and this does not).

Definition 3. *Let $K \subset \mathbf{C}$ be an effective field. Let C be a subset of $\mathbf{C}^{n-1}$ and $P \subset K[x_1, \ldots, x_{n-1}, x_n]$ be a finite set of polynomials whose main variable really is x_n. We say that P* separates above *C if for each $\alpha \in C$:*

1. *for each $p \in P$, the polynomial $\mathrm{lc}_{x_n}(p)$ does not vanish at α;*
2. *the polynomials $p(\alpha, x_n) \in \mathbf{C}[x_n]$, for all $p \in P$, are squarefree and coprime.*

Note that the empty set is trivially separable.

[5] `http://www.regularchains.org`

We then need an analogue of Definition 1 for the case of complex space. We follow [CM14a] and describe these (complex) cylindrical decompositions in terms of the tree data structure they are stored as.

Definition 4. *We define a* cylindrical decomposition *of* $\mathbf{C}^n$*, and its associated tree, by induction on* n*.*

Base Either: *There is one set* D_1*, the whole of* $\mathbf{C}$ *and* $\mathcal{D} = \{D_1\}$*;*
Base Or: *there are* r *non-constant square-free relatively prime polynomials* p_i *such that* D_i *is the set of zeros of* p_i*, and* D_{r+1} *is the complement:* $\{x : p_1(x)p_2(x)\cdots p_r(x) \neq 0\}$*:* $\mathcal{D} = \{D_1, \ldots, D_r, D_{r+1}\}$*.*
Base Tree: *The root and all the* D_i *as leaves of it.*
Induction: *Let* $\mathcal{D}'$ *be a cylindrical decomposition of* $\mathbf{C}^{n-1}$*. For each* $D_i \in \mathcal{D}'$*, let* r_i *be a non-negative integer, and* $P_i = \{p_{i,1}, \ldots, p_{i,r_i}\}$ *be a set of polynomials which separates over* D_i*.*
Induction Either: $r = 0$ *and we set* $D_{i,1} = D_i \times \mathbf{C}$*;*
Induction Or: *we set* $D_{i,j} = \{(\alpha, x) : \alpha \in D_i \wedge p_{i,j}(\alpha, x) = 0\}$*;*

$$D_{i,r+1} = \left\{(\alpha, x) : \alpha \in D_i \wedge \prod_j p_{i,j}(\alpha, x) \neq 0\right\};$$

Then: *a cylindrical decomposition of* $\mathbf{C}^n$ *is given by*

$$\mathcal{D} = \{D_{i,j} : 1 \leq i \leq |\mathcal{D}'|; 1 \leq j \leq r_i + 1\}.$$

Induction Tree: *If* T' *is the tree associated to* $\mathcal{D}'$ *then the tree associated to* $\mathcal{D}$ *is obtained by adding to each leaf* $D_i \in T'$ *as children all the* $D_{i,j}$ *such that* $1 \leq j \leq r_i + 1$*.*

Unlike Definition 1, the different roots of a given polynomial are not separated. Each cell is the zero set of a system of polynomial equations and inequations, where the main variables are all distinct: a triangular system [ALM99].

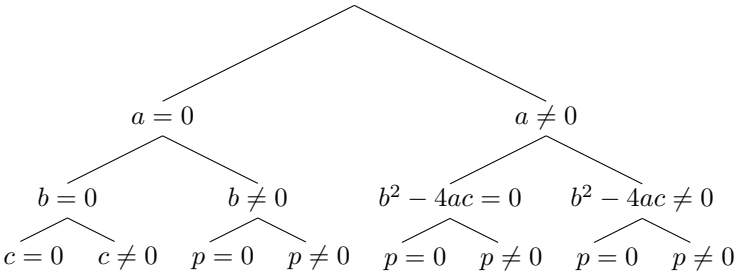
Definition 5. *Let* F *be a set of polynomials in* $k = K[x_1, \ldots, x_n]$*. A cylindrical decomposition* $\mathcal{D}$ *is* F-invariant *if, for each cell* $D \in \mathcal{D}$ *and each* $f_i \in F$*, either* f *vanishes at all points of* D *or* f *vanishes at no point of* D*.*

The trivial decomposition, obtained by taking the "either" branch each time, with one cell, is $\emptyset$-invariant. Given a cylindrical decomposition $\mathcal{D}$ which is F-invariant, and supposing $\widehat{F} = F \cup \{f\}$, [CM14a] shows how to refine $\mathcal{D}$ to a cylindrical decomposition $\widehat{\mathcal{D}}$ which is $\widehat{F}$-invariant, hence the "incremental" in the title of their paper. The key ingredients in this process are again leading coefficients, resultants and discriminants. The paper [CMXY09] shows, assuming that $K \subset \mathbf{R}$, how to construct from $\mathcal{D}$ a cylindrical algebraic decomposition of $\mathbf{R}^n$ which is sign-invariant for F.

The construction of the cylindrical decomposition can be seen, as pointed out in [CM14a], as an analogue of the projection phase of projection and lifting. Indeed, if n is small, it is often the case that the polynomials at level i in the tree corresponding to $\mathcal{D}$ are those in $\mathcal{P}_{n-i}$. The fundamental difference is that the $\mathcal{P}_i$ are *global* structures: over the whole cylindrical algebraic decomposition of $\mathbf{R}^k$ we need to isolate all the branches of all of $\mathcal{P}_{k+1}$, whereas there is a tree structure underpinning $\mathcal{D}$ and the cylindrical algebraic decomposition, which means that "polynomials are not considered when they are blatantly not relevant".

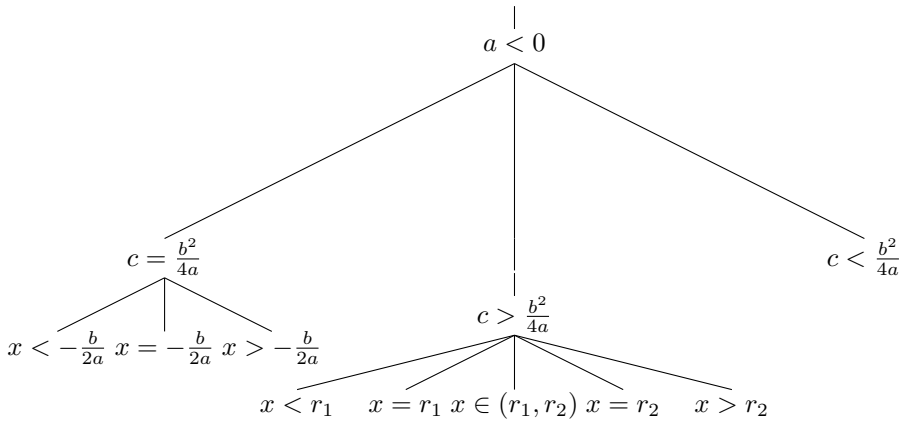
Example: Consider the parabola $p := ax^2 + bx + c$ and assume the variable ordering $x \succ c \succ b \succ a$. Suppose we were to use projection and lifting. Then the first projection identifies the coefficients a, b, c and the discriminant with respect to x: $b^2 - 4ac$. Subsequent projection do not identify any further projection polynomials for this example. Lifting produces CADs sign-invariant for these 4 projection polynomials, as well as p itself.

The regular chains approach would start by building the following tree, representing a cylindrical decomposition of $\mathbf{C}^n$:



This decomposition was produced to be sign-invariant for p. However, it does not insist on sign-invariance for the all the other projection polynomials. In particular, it is not sign-invariant for b. The polynomial b is included in the projection set because its vanishing can determine delineability, but only when the coefficient of the higher degree terms vanish. So, when $a = 0$ it is important to ensure b is sign-invariant, but not otherwise. Hence the tree above is doing only what is necessary to make the final conclusion about p.

The next step is to apply real root isolation, extending this tree to one representing a CAD. At the top level the case $a \neq 0$ must split into the two possibilities: $a < 0$ and $a > 0$. For brevity we display only the branch for $a < 0$ below (where r_1 and r_2 represent the two real roots of p in the case where the leading coefficient is negative and the discriminant positive). The full tree has 27 leaves, thus representing a CAD with 27 cells. This compares with a minimal CAD of 115 cells produced by projection and lifting to be sign invariant for all projection polynomials.



Are there significant savings in general? We refer the reader to [BDE+14, Table 1]. Here `PL-CAD` refers to our implementation of McCallum's algorithm of Section 2; `RC-INC-CAD` refers to the algorithm of [CM14a] (Section 3); and QEPCAD [Bro03] is another, highly optimised, implementation of McCallum's algorithm. Where both terminate, QEPCAD and `PL-CAD` often, though not always, have the same cell count. `RC-INC-CAD` does sometimes have the same count, but on other examples such as BC-Phisanbut-4, needs only 2007 cells, while both implementations of McCallum's algorithm need 51,763.

4 Quantifier Elimination

The original motivation for [Col75] was the following problem.

Problem 1 (Quantifier Elimination). Let $Q_i \in \{\exists, \forall\}$, and $\mathcal{L}_{\mathrm{RCF}}$ be the language of Boolean-connected equalities and inequalities concerning polynomials in $K[x_1, \ldots, x_n]$, where K is an effective field with $\mathbf{Q} \subseteq K \subset \mathbf{R}$. Given a statement (known as a Tarski statement, or a Tarski sentence if $k = 0$)

$$\Phi := Q_{k+1}x_{k+1} \ldots Q_n x_n \phi(x_1, \ldots, x_n) : \qquad \phi \in \mathcal{L}_{\mathrm{RCF}}, \tag{6}$$

the *Quantifier Elimination* problem is that of producing an equivalent

$$\Psi := \psi(x_1, \ldots, x_k) : \qquad \psi \in \mathcal{L}_{\mathrm{RCF}}. \tag{7}$$

In particular, $k = 0$ is a decision problem: is Φ true?

If we have a CAD $\mathcal{D}^{(n)}$ of $\mathbf{R}^n$ (noting that the x_i must be ordered in the same way in Definition 1 and formula (6)) sign-invariant for the polynomials of Φ, then constructing Ψ is conceptually easy.

1. The truth of ϕ in a cell $D_{\mathbf{i}}$ of $\mathcal{D}^{(n)}$ is that of ϕ at the sample point $s_{\mathbf{i}}$.
2. $\mathcal{D}^{(n)}$ projects to a CAD $\mathcal{D}^{(k)}$ of $\mathbf{R}^k$.
3. The truth of Φ in a cell $\widehat{D}_{\mathbf{j}} \in \mathcal{D}^{(k)}$ is then the appropriate ($\bigvee$ for $\exists$ etc.) Boolean combination of the truth of ϕ in the cells of $\mathcal{D}$ that project to $\widehat{D}_{\mathbf{j}}$.
4. Ψ is then the disjunction of the defining formulae for all the $\widehat{D}_{\mathbf{j}}$ for which Φ is true.

There is a problem in practice with the last step, first pointed out in [Bro99]. In the lifting stage, we produce branches θ_i of polynomials, with descriptions such as "that branch of $p(x_1, \ldots, x_l)$ which, above the sample point $s = (\alpha_1, \ldots, \alpha_{l-1})$, has the (unique) root in (β, γ)", and this is not a statement of $\mathcal{L}_{\mathrm{RCF}}$. We could equally describe it as "the third real branch of $p(x_1, \ldots, x_l)$ above s", but again this statement is not in $\mathcal{L}_{\mathrm{RCF}}$. Now by Thom's Lemma [CR88], we can describe this branch in terms of the signs of p *and its derivatives*, but, whereas these derivatives are in the Collins projection, they are not in the McCallum projection, or in the tree constructed by the method of Section 3. However, when it comes to describing $\widehat{D}_{\mathbf{j}}$, we can just add these (as described in [Bro99] for projection and lifting and in [CM14b] for regular chains CAD construction). The

additional cost is negligible, in particular, we do not need them for projection (Section 2), or for tree construction (Section 3).

Though it may depend non-linearly on polynomial degree etc., this process is linear in the number of cells in $\mathcal{D}^{(n)}$, and produces a disjunction of at most as many clauses as there are cells in $\mathcal{D}^{(k)}$.

5 Lower Bounds

This last remark is the basis of the complexity lower bounds in [DH88,BD07]. Both constructions use the fact that

$$\exists z_m \forall x_{m-1} \forall y_{m-1} \left(\begin{array}{c} (y_{m-1} = y_m \wedge x_{m-1} = z_m) \\ \vee\ (y_{m-1} = z_m \wedge x_{m-1} = x_m) \\ \Rightarrow y_{m-1} = F_{m-1}(x_m - 1) \end{array} \right) \tag{8}$$

encodes $y_m = F_{m-1}(F_{m-1}(x_m))$. Hence applying this construct $m-1$ times to $y_1 = F_1(x_1)$ gives

$$y_m = \underbrace{F_1(F_1(\cdots F_1}_{2^{m-1}\text{ times}}((x_n))\cdots).$$

This can then be used to produce expressions with n quantifiers and having $2^{2^{O(n)}}$ isolated point solutions, hence needing $2^{2^{O(n)}}$ cells to describe them (the $O(n)$ terms are $n/3 + O(1)$ in [BD07] and $n/5 + O(1)$ in [DH88]). An example which needs $2^{2^{O(n)}}$ cells for all possible variable orders is also produced in [BD07], along with another which needs $2^{2^{O(n)}}$ cells in one order, but a constant number in another. Hence the great interest in variable order selection methods for CAD [DSS04,EBDW14,HEW+14, to name a few].

The construction in (8) uses both $\exists$ and $\forall$ in a way that cannot be unnested. In fact, it is possible [Gri88] to decide Tarski sentences (i.e. no free variables) with a cost that is singly-exponential in n, but doubly-exponential in a, the number of *alternations* of $\exists$ and $\forall$ in (6). These methods, or any methods singly-exponential in n, have, in general, not been implemented, though there has been work on the purely existential case (for example [Hun08]).

6 Equational Constraints

The methods described in the previous sections produce decompositions which are sign- (or order-)invariant for a set of polynomials. In particular, we can apply steps 1–4 of Section 4 to the same CAD to solve (6) for any other ϕ involving the same polynomials. Indeed, as long as the x_i stayed in the same order, we could change the Q_i as well. [Col98] suggested that we could do better if ϕ was of the form $p_1 = 0 \wedge \phi'$, as we would not be interested in the behaviour of polynomials in ϕ' except when $p_1 = 0$. This was implemented in [McC99], who produced a CAD which was sign-invariant for p_1, and **when** $p_1 = 0$, sign-invariant for the polynomials in ϕ'. The main effect of this is to reduce the double exponent n

of m in (5) by 1, i.e. to take the square root of this term, as shown recently in [BDE+14] (14).

It is worth seeing how this works. Consider

$$\phi := (f_1 = 0) \wedge ((f_2 > 0) \vee (f_3 > 0)) . \tag{9}$$

Then a [McC84]-style projection ignoring the fact that there is an equation constraint would contain[6] three $\text{disc}(f_i)$ and three $\text{res}(f_i, f_j)$. However, [McC99] observes that we are not interested in f_2, f_3 except when $f_1 = 0$, and hence we need only consider $\text{disc}(f_1)$ and $\text{res}(f_1, f_2)$, $\text{res}(f_1, f_3)$, half as many polynomials.

Consider now

$$\phi_1 := ((g_1 = 0) \wedge (g_2 > 0)) \vee ((g_3 > 0) \wedge (g_4 = 0)) . \tag{10}$$

A [McC84]-style projection ignoring the fact that there is an equation constraint would contain four discriminants and six resultants. Although (10) does not contain an *overt* equational constraint, $\phi_1 \Rightarrow (g_1 = 0) \vee (g_4 = 0)$, which is $\phi_1 \Rightarrow (g_1 g_4 = 0)$, and so the equational constraint $g_1 g_4 = 0$ is implicit. If we study $g_1 g_4 = 0 \wedge \phi_1$ in the style of (9), and drop trivial resultants, we consider $\text{disc}(g_1 g_4)$, $\text{res}(g_1 g_4, g_2)$ and $\text{res}(g_1 g_4, g_3)$. Using the multiplicative properties of resultants and discriminants (which we would certainly do in practice!), this is $\text{disc}(g_1)$, $\text{disc}(g_4)$ and all the resultants except $\text{res}(g_2, g_3)$, i.e. two discriminants and five resultants.

Intuitively $\text{res}(g_1, g_3)$ and $\text{res}(g_2, g_4)$ are redundant, but how do we achieve this in general? This was solved in [BDE+13], where, rather than producing a sign-invariant CAD, we compute *truth table invariant* (a TTICAD) for the two propositions $(g_1 = 0) \wedge (g_2 > 0)$ and $(g_3 > 0) \wedge (g_4 = 0)$, i.e. on each cell, each of these two propositions is either identically true, or identically false. This process does indeed remove these two resultants, so we have two discriminants and three resultants.

Example: Consider (10) with

$$g_1 := x^2 + y^2 - 4, \qquad g_2 := (x-3)^2 - (y+3),$$

and

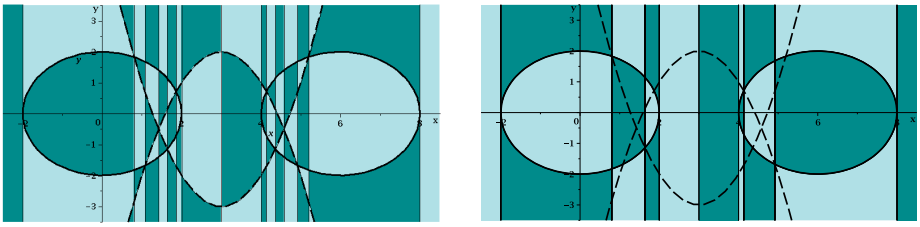
$$g_3 := (x-3)^2 + (y-2), \qquad g_4 := (x-6)^2 + y^2 - 4.$$

Figure 1 shows the two dimensional cells produced for both a sign-invariant CAD and a truth-table invariant CAD, built under ordering $x \prec y$. The sign-invariant CAD has 231 cells (72 full-dimensional but the splitting of the final cylinder is out of view) and the TTICAD 67 (22 full-dimensional).

By comparing the figures we see two types of differences. First, the CAD of the real line is split into fewer cells (there are not as many cylinders in $\mathbf{R}^2$). This is the effect of the reduction in projection polynomials identified, (less univariate

[6] It would also have some leading coefficients etc., but these are not the main drivers of the complexity in McCallum's projection.

Fig. 1. The left is a sign-invariant CAD, and the right a TTICAD, for (10) with the polynomials from the example.



polynomials with real roots to isolate). The second difference is that the full-dimensional cylinders are no longer split over the dashed lines. This came from an improvement in the lifting phase (discussed in detail in [BDE+14]). It leveraged the projection theory to conclude that we usually only need to lift with respect to equational constraints themselves.

More recently [BDE+14] truth-table invariance has been achieved even when there is no implicit equational constraint, as with an example of the form

$$((h_1 = 0) \wedge (h_2 > 0)) \vee (h_3 > 0). \tag{11}$$

The savings that can be achieved depend on the number of equational constraints involved in sub-clauses of the parent formula.

It is also possible to apply equational constraints in the regular chains technology view of CAD [CM14a], again even when there is no global equational constraint, as in (11) [BCD+14].

7 How Reliable is this?

Cylindrical algebraic decomposition can be used as tool in program verification, as in the MetiTarski tool [Pau12]. This leads to the question: who will verify the CAD, or at least the inferences we draw from it? We note that a positive answer to a purely existential question (equally, a negative answer to a purely universal question) is easily verified since we have a witness. The converse questions are essentially questions of refutation, see [JdM12]. Questions involving a mixture of quantifiers are much harder.

Almost all current implementations of CAD are based on computer algebra systems, which are generally unverified. We can at least compare, on a fairly level playing field, the implementations in Maple of four algorithms: see Table 1. The classification of the amount of mathematics involved is subjective, but we note that [McC84], and hence [BDEW13], relies on [Zar65,Zar75] to justify the smaller projection set compared with [Col75]. [CM14a] and [BCD+14] rely on, *inter alia*, [ALM99].

There are two challenges involved in verifying a CAD algorithm.

Table 1. Comparison of algorithms

Algorithm	Implementation	Code Lines (above Maple)	Specialist Mathematics
[Col75]	[EWBD14]	2600	some
[McC84]	[EWBD14]	2500	a lot
[CM14a]	[CM14a]	5000	medium
[BDEW13]	[EWBD14]	3000	a lot
[BCD+14]	[BCD+14]	5500	medium

1. There is a "program verification" question of ensuring that the algorithms produce the result that they say they will, i.e. that resultants, discriminants, real roots etc. are computed correctly. This is non-trivial, to say the least, sitting on top of an unverified computer algebra system, but should be feasible for an implementation based on a sound kernel, such as Coq or Isabelle.
2. There is a "mathematics verification" question whether the resulting decomposition is truly sign-/order-/truth table-invariant for the inputs. This is where the column labelled "Mathematics" in Table 1 comes in. The only attempt to produce verified CADs known to the authors, [CM12, in Coq], is based, not on [Col75] and its successors, but rather on [BPR06, chapter 2], itself essentially that of [Tar51].

2a. There is an interesting tension here between "precomputed" and *ad hoc* verification. An implementation based in [McC84] would essentially have to verify the relevant theorems from [Zar65,Zar75], but these could be imported as pre-verified lemmas. An implementation based on [CM14a] would verify that *in this case* we had an appropriate cylindrical decomposition of $\mathbf{C}^n$ which *in this case* translated to an appropriate cylindrical algebraic decomposition of $\mathbf{R}^n$.

8 Final thoughts

The topics we focussed on in this paper are implemented in Maple:

- CAD by Regular Chains is implemented in the RegularChains Library. A version of this ships with the core Maple distribution while the latest version is freely available from `http://www.regularchains.org/`.
- The authors' own work (equational constraints, truth-table invariance, sub-decompositions) is freely available in a Maple package ProjectionCAD. The latest version is available from: `http://opus.bath.ac.uk/43911/`.

Other implementations of cylindrical algebraic decomposition include:

- Mathematica [Str06]; The commands `CylindricalDecomposition` and `Reduce` can make use of an underlying CAD implementation. These commands can be exceptionally fast but it can be hard to judge the CAD components individually as they are just one of several underlying methods available and the output is in the form of formulae rather than cells.

- QEPCAD [Bro03]; a dedicated interactive command-line program available from `http://www.usna.edu/CS/qepcadweb/B/QEPCAD.html`. One notable feature is the SLFQ program which can simplify large quantifier free formulae giving more readable output. SAGE now has a QEPCAD interface.
- REDLOG [SS03]; this REDUCE package implements CAD along with other quantifier elimination methods such as virtual substitution.
- SYNRAC [IYAY13]; a MAPLE package notable for its symbolic-numeric approach. An older version is available for free download from: `http://jp.fujitsu.com/group/labs/en/techinfo/freeware/synrac/` with more recent advances part of the wider Todai Robot project.

The only reported experiments to cover all of these implementations were detailed in Section 4 of [BCD$^+$14].

Of course, this paper surveyed only a few of the recent advances in cylindrical algebraic decomposition. Others include (but are not limited to):

- The use of certified numerics in the lifting phase to minimise the amount of symbolic computation required [Str06,IYAY13].
- Local projection schemes [Str14], generic projection schemes [SS03] and single CAD cells [Bro13,JdM12].
- Problem formulation for CAD [DSS04,BDEW13,WEDB14] (projection and lifting) [EBDW14,EBC$^+$14] (regular chains). These all develop heuristics to help with choices, while [HEW$^+$14] applies machine learning in the form of support vector machines to pick a heuristic.
- Work on cylindrical algebraic sub-decompositions, which return only a subset of the cells in a full CAD [Sei06]. In [WBDE14] algorithms are given to return cells that lie on a prescribed variety, or have a designated dimension, while in [WDEB13] these techniques are combined to solve a motion planning problem. Note that if restricting to cells of full dimension then sample points can always be chosen to be rational, greatly reducing running time.

There are numerous unsolved problems, both theoretical and practical. Three that stand out to the authors are the following.

1. There is no complexity analysis of the Regular Chains method (though clearly it is subject to the lower bounds in Section 5).
2. There has been much progress in the last forty years, but implementations (at least for systems with alternations of quantifiers) are still doubly-exponential in the number of variables while the theory suggests we can do better.
3. Cylindricity is needed in step 3 of quantifier elimination, as $\exists$ translates into $\bigvee$ and $\forall$ into $\bigwedge$. However, in fact we only need this at the points where $\exists$ and $\forall$ alternate, so we can weaken the definition of cylindricity from being true for all π_k to merely being true for those k where x_k and x_{k+1} are governed by different quantifiers (or where x_k is unquantified but x_{k+1} is quantified, a concept we can call *block-cylindrical*. Unfortunately, we currently know of no way of computing a block-cylindrical algebraic decomposition without computing the full cylindrical algebraic decomposition first.

Acknowledgements

This work was supported by the EPSRC (grant number EP/J003247/1).

The authors thank Russell Bradford, Nicolai Vorobjov, David Wilson (University of Bath), Changbo Chen (Chinese Academy of Sciences, Chongqing), Zongyan Huang (University of Cambridge), Scott McCallum (Macquarie University) and Marc Moreno Maza (Western University).

References

[ALM99] P. Aubry, D. Lazard, and M. Moreno Maza. On the Theories of Triangular Sets. *J. Symbolic Comp.*, 28:105–124, 1999.

[AMW08] M. Achatz, S. McCallum, and V. Weispfenning. Deciding Polynomial-Exponential Problems. In D.J.Jeffrey, editor, *Proc. ISSAC 2008*, pages 215–222, 2008.

[BBDP07] J.C. Beaumont, R.J. Bradford, J.H. Davenport, and N. Phisanbut. Testing Elementary Function Identities Using CAD. *AAECC*, 18:513–543, 2007.

[BCD+14] R.J. Bradford, C. Chen, J.H. Davenport, M. England, M. Moreno Maza, and D.J. Wilson. Truth Table Invariant Cylindrical Algebraic Decomposition by Regular Chains. In *Proc. CASC 2014* (LNCS 8660), pages 44–58, 2014.

[BD07] C.W. Brown and J.H. Davenport. The Complexity of Quantifier Elimination and Cylindrical Algebraic Decomposition. In C.W. Brown, editor, *Proc. ISSAC 2007*, pages 54–60, 2007.

[BDE+13] R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson. Cylindrical Algebraic Decompositions for Boolean Combinations. In *Proc. ISSAC 2013*, pages 125–132, 2013.

[BDE+14] R.J. Bradford, J.H. Davenport, M. England, S. McCallum, and D.J. Wilson. Truth Table Invariant Cylindrical Algebraic Decomposition. http://arxiv.org/abs/1401.0645, 2014.

[BDEW13] R.J. Bradford, J.H. Davenport, M. England, and D.J. Wilson. Optimising Problem Formulation for Cylindrical Algebraic Decomposition. In J. Carette *et al.*, editor, *Proc. CICM 2013* (LNCS 7961), pages 19–34, 2013.

[BPR06] S. Basu, R. Pollack, and M.-F. Roy. Algorithms in Real Algebraic Geometry, 2nd ed. *Springer*, 2006.

[Bro99] C.W. Brown. Guaranteed Solution Formula Construction. In S. Dooley, editor, *Proc. ISSAC '99*, pages 137–144, 1999.

[Bro03] C.W. Brown. QEPCAD B: A program for computing with semi-algebraic sets using CADs. *ACM SIGSAM Bulletin 4*, 37:97–108, 2003.

[Bro05] C.W. Brown. The McCallum projection, lifting, and order-invariance. Technical report, U.S. Naval Academy, Computer Science Department, 2005.

[Bro13] C.W. Brown. Constructing a single open cell in a cylindrical algebraic decomposition. In *Proc. ISSAC '13*, pages 133–140. ACM, 2013.

[CM12] C. Cohen and A. Mahboubi. Formal Proofs in Real Algebraic Geometry: From Ordered Fields to Quantifier Elimination. *Logical Methods in Computer Science*, 8:1–40, 2012.

[CM14a] C. Chen and M. Moreno Maza. An Incremental Algorithm for Computing Cylindrical Algebraic Decompositions. In R. Feng *et al.*, editor, *Computer Mathematics*, pages 199—221. Springer Berlin Heidelberg, 2014.

[CM14b] C. Chen and M. Moreno Maza. Quantifier Elimination by Cylindrical Algebraic Decomposition Based on Regular Chains. In *Proc. ISSAC 2014*, pages 91–98, 2014.

[CMXY09] C. Chen, M. Moreno Maza, B. Xia, and L. Yang. Computing Cylindrical Algebraic Decomposition via Triangular Decomposition. In J. May, editor, *Proc. ISSAC 2009*, pages 95–102, 2009.

[Col75] G.E. Collins. Quantifier Elimination for Real Closed Fields by Cylindrical Algebraic Decomposition. In *Proceedings 2nd. GI Conference Automata Theory & Formal Languages*, pages 134–183, 1975.

[Col98] G.E. Collins. Quantifier elimination by cylindrical algebraic decomposition — twenty years of progess. In B.F. Caviness and J.R. Johnson, editors, *Quantifier Elimination and Cylindrical Algebraic Decomposition*, pages 8–23. Springer Verlag, Wien, 1998.

[CR88] M. Coste and M.-F. Roy. Thom's Lemma, the Coding of Real Algebraic Numbers and the Computation of the Topology of Semi-Algebraic Sets. *J. Symbolic Comp.*, 5:121–129, 1988.

[Dav85] J.H. Davenport. Computer Algebra for Cylindrical Algebraic Decomposition. Technical Report TRITA-NA-8511 NADA KTH Stockholm (Reissued as Bath Computer Science Technical report 88-10), 1985.

[Dav15] J.H. Davenport. Solving Computational Problems in Real Algebra/Geometry. To appear in *Annales Mathematicae et Informaticae*, 2015. http://opus.bath.ac.uk/42826/.

[DH88] J.H. Davenport and J. Heintz. Real Quantifier Elimination is Doubly Exponential. *J. Symbolic Comp.*, 5:29–35, 1988.

[DSS04] A. Dolzmann, A. Seidl, and Th. Sturm. Efficient Projection Orders for CAD. In J. Gutierrez, editor, *Proc. ISSAC 2004*, pages 111–118, 2004.

[EBC+14] M. England, R. Bradford, C. Chen, J.H. Davenport, M. Moreno Maza, and D. Wilson. Problem formulation for truth-table invariant cylindrical algebraic decomposition by incremental triangular decomposition. In S.M.Watt *et al.*, editor, *Proc. CICM 2014* (LNAI 8543), pages 45–60. Springer, 2014.

[EBDW14] M. England, R. Bradford, J.H. Davenport, and D.J. Wilson. Choosing a Variable Ordering for Truth-Table Invariant Cylindrical Algebraic Decomposition by Incremental Triangular Decomposition. In *Proc. ICMS 2014* (LNCS 8592), pages 450–457, 2014.

[EWBD14] M. England, D. Wilson, R. Bradford, and J.H. Davenport. Using the Regular Chains Library to build cylindrical algebraic decompositions by projecting and lifting. In *Proc. ICMS 2014* (LNCS 8592), pages 458–465, 2014.

[Gri88] D.Yu. Grigoriev. Complexity of Deciding Tarski Algebra. *J. Symbolic Comp.*, 5:65–108, 1988.

[GV88] D.Yu. Grigoriev and N.N. Vorobjov Jr. Solving Systems of Polynomial Inequalities in Subexponential Time. *J. Symbolic Comp.*, 5:37–64, 1988.

[HEW+14] Z. Huang, M. England, D. Wilson, J.H. Davenport, L.C. Paulson, and J. Bridge. Applying machine learning to the problem of choosing a heuristic to select the variable ordering for cylindrical algebraic decomposition. In S.M.Watt *et al.*, editor, *Proc. CICM 2014* (LNAI 8543), pages 92–107. Springer International, 2014.

[Hon90] H. Hong. An Improvement of the Projection Operator in Cylindrical Algebraic Decomposition. In S. Watanabe and M. Nagata, editors, *Proc. ISSAC '90*, pages 261–264, 1990.

[Hun08] G.B. Huntington. *Towards an efficient decision procedure for the existential theory of the reals*. PhD thesis, University of California at Berkeley, 2008.

[IYAY13] H. Iwane, H. Yanami, H. Anai, and K. Yokoyama. An effective implementation of symbolic-numeric cylindrical algebraic decomposition for quantifier elimination. *Theoretical Computer Science*, 479(0):43–69, 2013.

[JdM12] D. Jovanović and L. de Moura. Solving Non-Linear Arithmetic. In *Proc. IJCAR 2012*, pages 339–354, 2012.

[McC84] S. McCallum. *An Improved Projection Operation for Cylindrical Algebraic Decomposition*. PhD thesis, University of Wisconsin-Madison Computer Science, 1984.

[McC85] S. McCallum. An Improved Projection Operation for Cylindrical Algebraic Decomposition. Technical Report 548 Computer Science University Wisconsin at Madison, 1985.

[McC99] S. McCallum. On Projection in CAD-Based Quantifier Elimination with Equational Constraints. In S. Dooley, editor, *Proc. ISSAC '99*, pages 145–149, 1999.

[Pau12] L.C. Paulson. MetiTarski: Past and Future. In *Proc. Interactive Theorem Proving*, pages 1–10, 2012.

[Sei06] A. Seidl. Cylindrical Decomposition Under Application-Oriented Paradigms. PhD thesis (University of Passau, Germany), 2006.

[SS03] A. Seidl and T. Sturm. A generic projection operator for partial cylindrical algebraic decomposition. In *Proc. ISSAC '03*, pages 240–247, 2003.

[Str06] A. Strzeboński. Cylindrical algebraic decomposition using validated numerics. *J. Symbolic Computation*, 41(9):1021–1038, 2006.

[Str14] A. Strzeboński. Cylindrical algebraic decomposition using local projections. In *Proc. ISSAC '14*, pages 389–396. ACM, 2014.

[Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. 2nd ed., Univ. Cal. Press. Reprinted in *Quantifier Elimination and Cylindrical Algebraic Decomposition* (ed. B.F. Caviness & J.R. Johnson), Springer-Verlag, Wein-New York, 1998, pp. 24–84., 1951.

[Vor89] N.N. Vorobjov Jr. Deciding consistency of systems of polynomial in exponent inequalities in subexponential time. *Notes of Sci. Seminars of Leningrad Dept. of Math.Steklov Inst.*, 176, 1989.

[Vor92] N.N. Vorobjov Jr. The complexity of deciding consistency of systems of polynomial in exponent inequalities. *J. Symbolic Comp.*, 13:139–173, 1992.

[Wan00] D. Wang. Computing triangular systems and regular systems. *J. Symbolic Comp.*, 30(2):221–236, 2000.

[WBDE14] D. Wilson, R. Bradford, J.H. Davenport, and M. England. Cylindrical algebraic sub-decompositions. *Mathematics in Computer Science*, 8:263–288, 2014.

[WDEB13] D. Wilson, J.H. Davenport, M. England, and R. Bradford. A "piano movers" problem reformulated. In *Proc. SYNASC '13*, pages 53–60. IEEE, 2013.

[WEDB14] D. Wilson, M. England, J.H. Davenport, and R. Bradford. Using the distribution of cells by dimension in a cylindrical algebraic decomposition. In Proc. SYNASC '14. pages 53–60. IEEE. 2013

[Zar65] O. Zariski. Studies in equisingularity II. *Amer. J. Math.*, 87:972–1006, 1965.

[Zar75] O. Zariski. On equimultiple subvarieties of algebroid hypersurfaces. *Proc. Nat. Acad. Sci. USA*, 72:1425–1426, 3260, 1975.